



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/226,939	01/08/1999	JOHN K. VINCENT	346872000500	8916

23910 7590 04/17/2002

FLIESLER DUBB MEYER & LOVEJOY, LLP
FOUR EMBARCADERO CENTER
SUITE 400
SAN FRANCISCO, CA 94111

EXAMINER

LY, ANH

ART UNIT PAPER NUMBER

2172

DATE MAILED: 04/17/2002

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary

Application No.

09/226,939

Applicant(s)

VINCENT ET AL.

Examiner

Anh Ly

Art Unit

2172

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 04 March 2002.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 9-29 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 9-29 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
- Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
- 11) ☐ The proposed drawing correction filed on _____ is: a) ☐ approved b) ☐ disapproved by the Examiner.
- If approved, corrected drawings are required in reply to this Office action.
- 12) ☐ The oath or declaration is objected to by the Examiner.

Priority under 35 U.S.C. §§ 119 and 120

- 13) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. _____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- * See the attached detailed Office action for a list of the certified copies not received.
- 14) ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).
- a) ☐ The translation of the foreign language provisional application has been received.
- 15) ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Attachment(s)

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____
- 4) ☐ Interview Summary (PTO-413) Paper No(s). _____
- 5) ☐ Notice of Informal Patent Application (PTO-152)
- 6) ☐ Other:

DETAILED ACTION

Response to Arguments

1. Applicant's arguments filed on 03/04/2002, with respect to claims 9-29 have been considered but are moot in view of the new ground(s) of rejection.

Specification

2. 35 U.S.C. 112, first paragraph, requires the specification to be written in "full, clear, concise, and exact terms." The specification is replete with terms which are not clear, concise and exact. The specification should be revised carefully in order to comply with 35 U.S.C. 112, first paragraph. Examples of some unclear, inexact or verbose terms used in the specification are: page 2, lines 3, "US Patent No. 6,651,111", replace with --US Patent No. 5,651,111--.

Claim Rejections - 35 USC § 112

3. Claim 9 is rejected under 35 U.S.C. 112, first paragraph, as containing a single step/means.

2164.08(a) Single Step Claim:

A single Step claim, i.e., where a means recitation does not appear in combination with another recited element of means, is subject to an undue breadth rejection under 35 U.S.C. 112, first paragraph. In re Hyatt, 708 F.2d 712, 714-715, 218 USPQ 195, 197 (Fed. Cir. 1983) (A single means

Art Unit: 2172

claim which covered every conceivable means for achieving the stated purpose was held nonenabling for the scope of the claim because the specification disclosed at most only those means known to the inventor. When claims depend on a recited property, a fact situation comparable to Hyatt is possible, where the claim covers every conceivable structure (means) for achieving the stated property (result) while the specification discloses at most only those known to the inventor.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

5. Claims 9, 17, 22 and 27-29 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 4,829,427 issued to Green.

With respect to claim 9, Green discloses dependency querying a database catalog for direct dependencies of a code object and then for each dependency found, doing the query recursively until all basic dependencies are generated into a dependency tree graph as claimed (col. 6, lines 5-34, col. 7, lines 58-67, col. 8, lines 1-25 and col. 9, lines 4-41; col. 14, lines 11-60 and col. 19, lines 21-30).

Green does not clearly disclose "direct dependencies of code object and a dependency tree." But, however, Green teaches code generators have the same dependencies as optimizers (col. 1, lines 32-67). Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Green such as code generator so as to obtain a method of generating a basic dependency tree of a code object in the recursive path analysis DBMS procedures environment.

With respect to claim 17, Green discloses using a recursive algorithm for querying a database catalog for direct dependencies of a code object and then for each dependency found, doing the query recursively until all basic dependencies are generated into a dependency graph; using a parser on each of the code objects in the dependency graph to identify DML statements (see fig. 12, col. 5, lines 25-67 and col. 6, lines 1-45) that "fire" triggers so as to identify dependencies on triggers; applying the recursive algorithm on each of the triggers to incorporate the dependencies of the triggers into the dependency graph; and repeating steps 1-3 for incorporating dependencies on triggers and their dependencies until new dependencies are not added to the dependency graph (col. 6, lines 5-34, col. 7, lines 58-67, col. 8, lines 1-25 and col. 9, lines 4-41; col. 14, lines 11-60 and col. 19, lines 21-30).

Green does not clearly disclose "direct dependencies of code object and a dependency tree." But, however, Green teaches code generators have the same dependencies as optimizers (col. 1, lines 32-67). Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Green such as code generator so as to obtain a

Art Unit: 2172

method of generating a basic dependency tree of a code object in the recursive path analysis DBMS procedures environment.

With respect to claim 22, Green discloses applying a recursive algorithm that queries a database for dependency information and outputs a direct dependency graph of a database code object, the "direct dependency graph" containing dependencies that do not involve dependencies on triggers and on implementations of object oriented code objects in the database; applying the recursive algorithm on each of the object oriented code objects in the dependency graph to incorporate dependencies of implementations of code objects in the database; and repeating steps 1 and 2 for incorporating dependencies of implementation of object oriented code objects until new dependencies are not added to the dependency graph (col. 6, lines 5-34, col. 7, lines 58-67, col. 8, lines 1-25 and col. 9, lines 4-41; col. 14, lines 11-60 and col. 19, lines 21-30; see fig. 12).

Green does not clearly disclose "direct dependencies of code object and a dependency tree." But, however, Green teaches code generators have the same dependencies as optimizers (col. 1, lines 32-67). Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Green such as code generator so as to obtain a method of generating a basic dependency tree of a code object in the recursive path analysis DBMS procedures environment.

With respect to claim 27, Green discloses a digital computer (col. 6, lines 5-21); a database server coupled to the computer (abstract and col. 1, lines 15-66); a data base coupled to the database server having data stored therein, the data including object oriented code objects, specifications of packages, implementations of packages, specifications of types, implementations of types and triggers (col. 1, lines 15-66, col. 3, lines 32-55 and col. 6, lines 5-21); and a code mechanism for generating a dependency graph, the dependency graph being a data structure and having entries to contain representations of depending code objects, of packages, implementations of

Art Unit: 2172

packages, specifications of types, implementations of types, triggers and dependencies of triggers which are relevant to the target data base code object (col. 6, lines 5-34, col. 7, lines 58-67, col. 8, lines 1-25 and col. 9, lines 4-41; col. 14, lines 11-60 and col. 19, lines 21-30; see fig. 12).

Green does not clearly disclose "direct dependencies of code object and a dependency tree." But, however, Green teaches code generators have the same dependencies as optimizers (col. 1, lines 32-67). Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Green such as code generator so as to obtain a method of generating a basic dependency tree of a code object in the recursive path analysis DBMS procedures environment.

With respect to claim 28, Green discloses providing a data base coupled to the computer having data stored therein, the data including representations of object oriented code objects, specifications of packages, implementations of packages, specifications of types, implementations of types and triggers; and using a recursive code mechanism for generating a dependency graph, the dependency graph being a data structure and having entries to contain representations of dependent code objects, specifications of packages, implementations of packages, specifications of types, implementations of types, triggers and dependencies of triggers which are relevant to the target data base code object (col. 6, lines 5-34, col. 7, lines 58-67, col. 8, lines 1-25 and col. 9, lines 4-41; col. 14, lines 11-60 and col. 19, lines 21-30; see fig. 12).

Green does not clearly disclose "direct dependencies of code object and a dependency tree." But, however, Green teaches code generators have the same dependencies as optimizers (col. 1, lines 32-67). Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to employ the teachings of Green such as code generator so as to obtain a method of generating a basic dependency tree of a code object in the recursive path analysis DBMS procedures environment.

Art Unit: 2172

6. Claims 10-16, 18-21, 23-26 and 29 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 4,829,427 issued to Green in view of US Patent No. 5,651,111 issued to McKeeman et al. (hereinafter as McKeeman).

With respect to claim 10, Green discloses a method of generating code as discussed in claim 9.

Green does not explicitly indicate, "using the generated dependency tree to compile code objects in debug mode as part of a database code object debugging tool."

However, McKeeman discloses a debugging tool as claimed (abstract, col. 7, lines 32-51, col. 16, lines 45-67 and col. 19, lines 22-52).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Green with the teachings of McKeeman so as to have a method of generating a basic dependency tree of a code object tool because the combination would provide the method that drive the debugging tool and testing software and provide a more efficient means of testing software, insure higher quality software due to more complete and sufficient testing, and save software development time and computer resources (McKeeman – col. 3, lines 6-40) in the in the recursive path analysis DBMS procedures environment.

With respect to claims 11-16, Green discloses a method of generating a basic dependency tree of a code object as discussed in claim 9.

Green does not explicitly indicate, "using the generated dependency tree to identify calling paths in a database code object coverage tool; using the generated dependency tree to identify calling paths in a database code object profiling tool; using the generated dependency tree in a database code object testing tool; using the generated dependency tree to identify dependent that are INVALID in the database; using the generated dependency tree to identify cyclic dependencies among database code objects; using the generated dependency tree in dependency graph presentation tool."

However, Mckeeman discloses code coverage tool as claimed (col. 7, lines 14-51, col. 18, lines 60-67 and col. 19, lines 1-4); profiling tool as claimed (col. 19, lines 1-4); testing tool as claimed (col. 19, lines 5-40); INVALID values as claimed (col. 5, lines 38-48); to identify cyclic dependencies as claimed (col. 6, lines 60-67 and col. 7, lines 1-8); and graphic routine as claimed (col. 14, lines 21-32).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Green with the teachings of McKeeman so as to have a method of generating a basic dependency tree of a code object tool because the combination would provide the method that drive the debugging tool and testing software and provide a more efficient means of testing software, insure higher quality software due to more complete and sufficient testing, and save software development time and computer resources (McKeeman – col. 3, lines 6-40) in the in the recursive path analysis DBMS procedures environment.

With respect to claims 18-21, Green discloses a method of generating a basic dependency tree of a code object as discussed in claim 17.

Green does not explicitly indicate, “using the generated dependency tree to identify calling paths in a database code object coverage tool; using the generated dependency tree to identify calling paths in a database code object profiling tool; using the generated dependency tree in a database code object testing tool; using the generated dependency tree to identify dependent that are INVALID in the database.”

However, Mckeeman discloses code coverage tool as claimed (col. 7, lines 14-51, col. 18, lines 60-67 and col. 19, lines 1-4); profiling tool as claimed (col. 19, lines 1-4); testing tool as claimed (col. 19, lines 5-40); INVALID values as claimed (col. 5, lines 38-48).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Green with the teachings of McKeeman so as to have a method of generating a basic dependency tree of a code object tool because the combination would provide the method that drive the debugging tool and testing software and provide a more efficient means of testing software, insure higher quality software due to more complete and sufficient

testing, and save software development time and computer resources (McKeeman – col. 3, lines 6-40) in the in the recursive path analysis DBMS procedures environment.

With respect to claims 23-26, Green discloses a method of generating a basic dependency tree of a code object as discussed in claim 22.

Green does not explicitly indicate, “using the generated dependency tree to identify calling paths in a database code object coverage tool; using the generated dependency tree to identify calling paths in a database code object profiling tool; using the generated dependency tree in a database code object testing tool; using the generated dependency tree to identify dependent that are INVALID in the database.”

However, Mckeeman discloses code coverage tool as claimed (col. 7, lines 14-51, col. 18, lines 60-67 and col. 19, lines 1-4); profiling tool as claimed (col. 19, lines 1-4); testing tool as claimed (col. 19, lines 5-40); INVALID values as claimed (col. 5, lines 38-48).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Green with the teachings of McKeeman so as to have a method of generating a basic dependency tree of a code object tool because the combination would provide the method that drive the debugging tool and testing software and provide a more efficient means of testing software, insure higher quality software due to more complete and sufficient testing, and save software development time and computer resources (McKeeman – col. 3, lines 6-40) in the in the recursive path analysis DBMS procedures environment.

With respect to claim 29, Green discloses a recursive code mechanism for generating a dependency graph of the target data base code object the dependency graph being a data structure and having entries to contain representations of depending code objects, specifications of packages, implementations of packages, triggers and dependencies of triggers which are relevant to the target data base code object; and a program code mechanism for using the dependency graph to debug the target data base code object.

Green does not explicitly indicate. “using the dependency graph to debug the target data base code object.”

Art Unit: 2172

However, McKeeman discloses a debugging tool as claimed (abstract, col. 7, lines 32-51, col. 16, lines 45-67 and col. 19, lines 22-52).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to combine the teachings of Green with the teachings of McKeeman so as to have a method of generating a basic dependency tree of a code object tool because the combination would provide the method that drive the debugging tool and testing software and provide a more efficient means of testing software, insure higher quality software due to more complete and sufficient testing, and save software development time and computer resources (McKeeman – col. 3, lines 6-40) in the in the recursive path analysis DBMS procedures environment.

Contact Information

7. Any inquiry concerning this communication should be directed to Anh Ly whose telephone number is (703) 306-4527. The examiner can be reached on Monday - Friday from 8:00 AM to 4:00 PM.

If attempts to reach the examiner are unsuccessful, see the examiner's supervisor, Kim Vu, can be reached on (703) 305-4393.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to:

(703) 746-7238 (for After Final communications intended for entry)

or:

(703) 746-7239 (for formal/Official communications intended for entry)

Art Unit: 2172

or:

(703) 746-7240 (for informal or draft communications or Customer Service Center , please label "PROPOSED" or "DRAFT")


Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington, VA, Fourth Floor (receptionist).

Inquiries of a general nature or relating to the status of this application should be directed to the Group receptionist whose telephone number is (703) 305-9600.

AL



Apr. 12th, 2002.



JEAN M. CORRIELUS
PRIMARY EXAMINER